Theory of Computer Science

G. Röger | University of Basel
Spring Term 2019 | Computer Science

Exercise Sheet 5 — Solutions

Exercise 5.1 (Regular Expressions; 2 Points)

Consider the following regular expressions over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. For each regular expression, specify two words that are in the corresponding language and two words that are not in the corresponding language.

(a) $\mathtt{bba}|\mathtt{bbb}$

(b) $\mathtt{b}^*\mathtt{a}(\mathtt{b}^*\mathtt{ab}^*\mathtt{ab}^*)^*$

(c) $(\mathtt{a}(\mathtt{a}|\mathtt{b})|\mathtt{b})(\mathtt{a}|\mathtt{b})^*$

(d) $(\varepsilon|\mathtt{a})\mathtt{b}|\mathtt{b}\emptyset\mathtt{a}$

Solution:

(a) $\mathcal{L}(\mathtt{bba}|\mathtt{bbb}) = \{\mathtt{bba}, \mathtt{bbb}\}$
The words bba and bbb are in the language. Examples for words that are not in the language are a and bbabb.

(b) $\mathcal{L}(\mathtt{b}^*\mathtt{a}(\mathtt{b}^*\mathtt{ab}^*\mathtt{ab}^*)^*) = \{w \in \Sigma^* \mid w \text{ contains an odd number of } \mathtt{a}\text{s}\}$
Examples for words in the language are a and ababbbaaa. Examples for words that are not in the language are aa and abbbba.

(c) $\mathcal{L}((\mathtt{a}(\mathtt{a}|\mathtt{b})|\mathtt{b})(\mathtt{a}|\mathtt{b})^*) = \Sigma^* \setminus \{\varepsilon, \mathtt{a}\}$
The words b and ab are examples for words in the language. The words that are not in the language are ε and a.

(d) $\mathcal{L}((\varepsilon|\mathtt{a})\mathtt{b}|\mathtt{b}\emptyset\mathtt{a}) = \{\mathtt{b}, \mathtt{ab}\}$
The words b and ab are in the language. Examples for words that are not in the language are ε and ba.

Exercise 5.2 (Pumping Lemma for Regular Languages; 4 Points)

Are the following languages over $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ regular? If so, prove it by specifying a regular expression which describes the language. If not, prove it with help of the Pumping-Lemma.

(a) $L_1 = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^{n+m} \mid m, n \in \mathbb{N}_0\}$

Solution:

Assume L_1 is regular. Let p be a pumping number of L_1. The word $x = \mathtt{a}^p\mathtt{b}^3\mathtt{c}^{p+3}$ is in L_1 and satisfies $|x| \geq p$. We know from the pumping lemma that there are words u, v and w with $x = uvw$, $|uv| \leq p$, $|v| \geq 1$ and $uv^iw \in L_1$ for all $i \geq 0$.

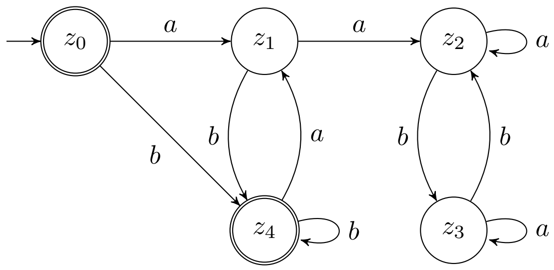
From $|uv| \leq p$ we know that uv can only consist of as. If we pump x smaller, i.e. we choose $i = 0$, we get the word $x_0 := uv^0w = \mathtt{a}^{p-|v|}\mathtt{b}^3\mathtt{c}^{p+3}$. As $|v| \geq 1$ we know that $p-|v|+3 < p+3$ and see that $x_0 \notin L_1$ (since the number of as added to the number of bs is not the number of cs, and thus the properties of the language are not satisfied). This is a contradiction to the pumping lemma and thus L_1 cannot be regular.

(b) $L_2 = \{\mathtt{a}^n\mathtt{b}^3\mathtt{c}^m\mathtt{d}^3 \mid m, n \in \mathbb{N}_0\}$

Solution:

L_2 is regular because it is described by the regular expression $\mathtt{a}^*\mathtt{bbbc}^*\mathtt{ddd}$.

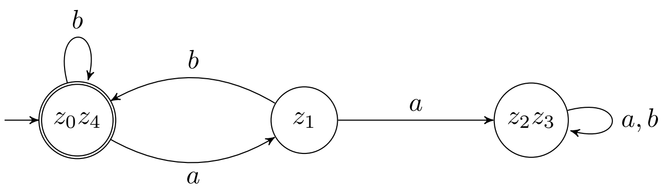
Exercise 5.3 (Minimal DFA; 2 Points)
Specify a minimal DFA which is equivalent to the following DFA:



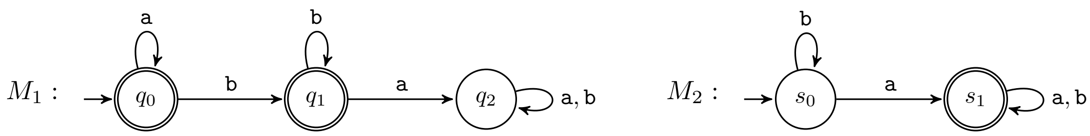
Solution:
We use the algorithm on slide 7 of slide set C4 to construct the minimal DFA.

	z_0	z_1	z_2	z_3
z_1	×			
z_2	×	×		
z_3	×	×		
z_4		×	×	×

We can merge node z_0 with z_4 and node z_2 with z_3 to get a minimal DFA.

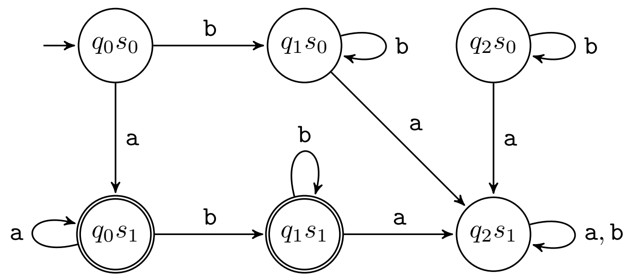


Exercise 5.4 (Product Automaton; 2 Points)
Consider the following DFAs M_1 and M_2.



Specify the product automaton that accepts $\mathcal{L}(M_1) \cap \mathcal{L}(M_2)$.
How would you have to change the definition of the end states (in general) to receive an DFA for the union of two languages?

Solution:
$\mathcal{L}(M_1) = \{\mathtt{a}^m\mathtt{b}^n \mid m, n \geq 0\}$, $\mathcal{L}(M_2) = \{w \mid w \text{ contains at least one } \mathtt{a}\}$
$\mathcal{L}(M_1) \cap \mathcal{L}(M_2) = \{a^m b^n \mid m \geq 1, n \geq 0\}$
The product automaton looks as follows:



For the intersection of the two languages, a state of the product automaton is an end state if it combines two end states of the two original DFAs. If we only required that one of the states in the pair must be an end state of an original DFA, the product automaton would accept the union of the languages.